EXPRESS MAIL NO.: EL 662 092 595 US

International Business Machines Corporation Docket No.:
BLD9-2000-0058US1

Ohlandt, Greeley, Ruggiero & Perle, L.L.P. Docket No.: 963.0046USU

Patent Application Papers of: J. Bruce Mixer, Jr.

# A METHOD AND APPARATUS FOR USING PRINT JOBS FOR DOWNLOADING AND EXECUTING PRINTER CONTROLLER CODE

## FIELD OF THE INVENTION:

This invention pertains to downloading program code to a device through an interface to replace or add to present microcode.

## BACKGROUND OF THE INVENTION:

As electronic devices have become more sophisticated, they now typically include computing components such as a processor, volatile memory, non-volatile memory and at least one interface to other components of the device. Microcode for directing the activity of the processor is usually resident in the non-volatile memory, and is typically loaded into the volatile memory and executed by the processor to perform various activities and operations. Computer peripherals, such as serial ports, USB hubs, modems, disk drives, network interface cards, and printers are especially suited for these types of components.

Supporting computer peripherals in the field, especially printers, can be difficult because, in many cases, specialized routines must be loaded into the printer in order to diagnose problems.

Reference in this regard may be had to US Patent No. 5,768,495, entitled "Method and Apparatus for Printer Diagnostics," by Campbell et al., issued June 16, 1998.

Supporting the microcode of a printer at a customer site is also a difficult process because, more often than not, diagnosing and correction of microcode problems requires a

customer engineer to be physically present at the printer. This is a disadvantage in that it results in a higher support cost because of travel time, availability of tools, etc. In order to install new microcode, the new microcode must be downloaded into the non-volatile memory of the device. This may require a special hardware and/or software interface. In most cases, the operating system of the computer controlling the printer controls all access to the printer and only provides a user with operations related to the function of the device, for example, printing. In this example, the computer would provide no facility to download anything other than print jobs to the printer. When the operating system permits no other access, a special application running under the operating system may be required to download new program code. Utilizing such an application involves obtaining the application, installing the application, and invoking the application in order to perform the installation.

To further complicate matters, the operating system may not relinquish control of the printer to the application and may have to be disabled or shut down in order to run the application for performing such a download.

Other techniques include embedding a special code within a data field of a known control code for the particular peripheral. The data field and the special code may be chosen such that the possibility of the special code being used as actual data in that data field is extremely low. Upon recognizing the control code coupled with the special code in the data field, the peripheral waits for a specific second command. If the specific second command is received the peripheral loads the data attached to that subsequent command as new microcode, and reboots.

Reference in this regard may be had to US Patent No. 5,659,801, entitled, "Method and Apparatus for Replacing Resident Peripheral Device Control Microcode by Download via an Application Program," by W. Kopsaftis, issued August 19, 1997.

OBJECTS AND ADVANTAGES OF THE INVENTION:

It is a first object and advantage of this invention to provide a method and apparatus for downloading microcode to a device without requiring the use of specialized downloading

hardware or application routines.

It is a further object and advantage of this invention to download microcode to a device by delivering it in a format that the device is already adapted to receive.

It is still further an object and advantage of this invention to download microcode to a device by downloading an executable program to the device in a format that the device is already adapted to receive, and then running the executable program.

## SUMMARY OF THE INVENTION

The foregoing and other problems are overcome and the objects of the invention are realized by methods and apparatus in accordance with embodiments of this invention.

A method for updating microcode of a printer is disclosed which includes the steps of embedding a microcode update file as a module in a print job file, inputting the print job file to the printer, recognizing that the print job file includes the microcode update file, and writing the microcode update file to a memory area in the printer indicated in the print job file. The microcode update file may include an executable program and after writing the microcode update file to memory execution may be transferred to the executable program.

## BRIEF DESCRIPTION OF THE DRAWINGS

The above set forth and other features of the invention are made more apparent in the ensuing Detailed Description of the Invention when read in conjunction with the attached Drawings, wherein:

Figure 1 is block diagram of a printer in accordance with the teachings herein;

Figure 2 is a block diagram of the system architecture of the printer shown in Figure 1;

Figure 3 shows a block diagram of a portion of the system non-volatile memory implemented as one or more storage devices.

Figure 4 shows a diagram of the structure of a print job file for downloading a microcode update file in accordance with these teachings;

Figure 5 shows a diagram of the file header of the print job file for downloading a microcode update;

Figure 6 shows a status page, printed as a result of a microcode update;

Figure 7 shows a diagram of the module header of a module within a microcode update; and

Figure 8 shows a flow chart of the steps for preparing, downloading and utilizing a microcode update.

## DETAILED DESCRIPTION OF THE INVENTION

Figure 1 shows a diagram of a printer 10 in accordance with the teachings of this invention. The printer includes an enclosure 15, an output tray 20, media trays 25, 30 for holding a quantity of media to be printed upon, a front panel 35, including a display 37 and controls 40 for operating the printer. The front panel 35 typically includes status indicators 45 for indicating the status of the various processes within the printer 10.

Figure 2 shows a schematic diagram of the system architecture 47 of the printer 10. The printer 10 includes a processor 50 for controlling the operation of the printer, and volatile memory 55 and non-volatile memory 60 for storing programs. The printer 10 preferably also includes a printing engine 65, a user interface 70 for receiving user inputs and for displaying messages and status to a user, and an input output (I/O) interface 75 for receiving print job files. The I/O interface 75 is also capable of receiving, through links 85 or 92, a computer readable medium embodied in a carrier wave of a propagated signal, 93, such as an internet transmission, where the computer readable medium carries one or more instructions for the processor 50. The processor 50, volatile memory 55, non-volatile memory 60, printing engine 65, user interface 70, and I/O interface 75 all communicate through bus 77.

The volatile memory 55 includes memory for executing programs, referred to herein as

execution memory 57.

The non-volatile memory 60 may have a portion configured as a boot ROM 61. Referring also to Figure 3, the non-volatile memory 60 preferably includes memory 62 implemented as one or more storage devices $63_1 \ldots 63_n$. The storage devices $63_1 \ldots 63_n$ may include reprogrammable memory devices such as EPROM, EEPROM, compact flash memory, etc. The storage devices $63_1 \ldots 63_n$ may further include one or more hard drives or other types of storage media. The non-volatile memory may also be distributed throughout other parts of the system. For example, a portion of the non-volatile memory may include flash memory 67 resident on the I/O interface 70. As mentioned above, the non-volatile memory 60 includes programs, and in particular includes microcode for directing the activities of the processor, including receiving a print job file through the I/O interface 75 and for utilizing the information in the print job file to direct the printing engine.

For purposes of this invention, microcode includes the following: CTL code, that is, printer control code which includes routines called by the operating system for accessing and controlling the interaction between the various components of the printer 10; IPDS code, a higher level printer language code for directing the printer engine; flash resource code, that is any code to be stored in flash memory; and, hard disk resource code, that is, any code to be stored on a hard disk. Microcode may also include; twinax NIC code which is code to be stored in non-volatile memory 68 of I/O Interface 75 (Figure 2) when the I/O Interface 75 is what is commonly referred to as a twinaxial network interface card; and, coax NIC code which is code to be stored in non-volatile memory 68 of I/O Interface 75 (Figure 2) when the I/O Interface 75 is what is commonly referred to as a coaxial network interface card. In addition, microcode may further include operating system code, code that provides special features for the printer 10, web pages that may be present in the printer 10, as well as data files such as fonts.

As shown in Figure 3, each storage device $63_1 \ldots 63_n$ may include a hierarchical file structure comprising one or more directories $64_1 \ldots 64_n$, each capable of containing one or more subdirectories $65_1 \ldots 65_n$.

Returning to Figure 2, an external computing device 80 may be connected to the printer

10 through the I/O interface 75 either by way of a link 85, or by way of a links 90, 92 and a network 95. The network 95 may include the PSTN, the internet, a wireless network, a wired network, and may further include other types of networks including X.25, TCP/IP, ATM, etc., as long as the network is suitable for conveying information to said printer 10. The computing device 80 preferably includes the capability to generate print job files to update microcode in printer 10, and may further include the ability to generate print job files to be printed. The I/O interface 70 may include a network interface, a serial or parallel interface, or any other type of interface, which is suitable for conveying information to be printed to the printer 10. This information is delivered in a particular format known as a print job file 97 (Figure 4).

In accordance with the teachings of this invention a print job file may also include new microcode, also referred to as a microcode update, for updating or extending the current microcode in the non-volatile memory 60. The current microcode in the non-volatile memory 60 is assumed to include routines for receiving a print job file 97 into volatile memory 55, recognizing the print job file 97 as including a microcode update, and then performing operations based on information in the microcode update.

Figure 4 shows a diagram of the structure of the print job file 97 in accordance with the teachings of the invention. The file header 100 includes information indicating that the file 97 includes program code. The print job file 97 also includes one or more modules $105_1$-$105_n$ which include the microcode update. Each module $105_1$-$105_n$ has a corresponding module header $110_1$-$110_n$ which contains additional information about the particular module, including a destination storage device 63 and directory 64 within non-volatile memory 60, that is, the additional information designates a location within non-volatile memory 60 for storing the module upon downloading.

Figure 5 shows a diagram of a presently preferred embodiment of the file header 100 for the print job file 97. The first nine bytes 115 of the file header 100 include a known indicator of the start of the print job file known as "User Exit Language." The next seven bytes 120 are known as lead-in and include three hexadecimal characters and the four letters "C", "O", "D", and "E". Upon identifying the first 16 bytes as having this particular structure, the processor recognizes that a microcode update file is included in

the print job file 97, invokes a program or task to control updating the microcode and passes the print job file content to that task. The next nineteen bytes 123, that is, bytes 16-34 of the file header 100, include the version of the microcode update file being downloaded. Bytes 35-38, designated 125 in Figure 5, include the number of modules in the print job file 97. The next byte 130 contains an identifier for the printer that the microcode update file is designed for. Bytes 40-41 (135) include an indicator of the memory size required for the print job file 97.

Still referring to Figure 5, the file header 100, in bytes 42-122 (165) includes 81 ASCII characters for the upper status page descriptor 145 (Figure 6), and in bytes 123-203 (170) includes 81 ASCII characters for the lower status page descriptor 160. The next 31 bytes 175 of the file header 100 include a description of the type of microcode included in the microcode update. The types of acceptable descriptions include the types of microcode described above, that is, CTL code, IPDS code, flash resource code, hard disk resource code, twinax NIC code, and coax NIC code. The following eight bytes 180, that is, bytes 235 to 242, include a random signature number for identification of the microcode update. Bytes 243-246 (185) include a checksum of all previous bytes in the file header 100. Byte 247 (190) designates suppressing printing of a status page previously discussed if an ASCII "S" is present in the byte.

When a microcode update file has been downloaded and processed, as will be explained below, the processor 50 may cause a status page 140 to be printed as shown in Figure 6. The status page 140 includes an upper status page descriptor 145, the status of the update operation 150, the original versions of certain portions of the microcode 155, the current (after updating) versions of those certain portions of the microcode, and a lower status page descriptor 160.

When a microcode update file has been downloaded and processed, as will be explained below, the processor 50 may cause a status page 140 to be printed as shown in Figure 6. The status page 140 includes an upper status page descriptor 145, the status of the update operation 150, the original versions of certain portions of the microcode 155, the current (after updating) versions of those certain portions of the microcode, and a lower status

page descriptor 160.

Returning to Figure 5, byte 248 (195) of the file header 100 includes bits according to the teachings of the invention that indicate how the print job file 97 including the microcode update, will be processed. The operation of these bits will be described in detail below.

Bytes 249 through 254 (200) may include optional data, and byte 255(205) is a lead out string indicating the end of the file header 100.

Each individual module includes its own header 110, as shown in Figure 7. The first six bytes 210 of the module header 110 include the word "MODULE," designating that this portion of the print job file 97 is a module. The next 31 bytes 215, that is, bytes 6-36, include the name of the module in ASCII (with no path information). Bytes 37-55 (220) include the version of the module, also in ASCII. The following four bytes 225 indicate the date the module was released, formatted in seconds from January 1, 1970. Bytes 60-100 (230) specify the target location path, that is, the storage device 63, directory 64, and subdirectory 65 (if applicable) in non-volatile memory 60 where the module is to be stored. Byte 101 (235) includes bit flags that define various operations to be performed in conjunction with the particular module. A bit flag of 01 hex directs the current microcode to uncompress the module. A bit flag of 02 hex directs the current microcode to create a file using the target location path 230 as the path and filename. A bit flag of 04 hex directs the current microcode to delete the file found at the target location path 230. A bit flag of 08 hex directs the current microcode to create a directory using the target location path 230. A bit flag of 10 hex directs the current microcode to delete the directory found at the target location path 230. A bit flag of 20 hex directs the current microcode to deposit the corresponding module body 105 at a default location, known to the system, in non-volatile memory 68 of the I/O interface 75, when the I/O interface is a coaxial or triaxial network interface card.

Bytes 102-105 (240) of the module header 110 include the total number of bytes that the module encompasses, excluding the module header 110. The following four bytes 245 indicate a checksum for the module. The checksum is preferably calculated as a 16 bit truncated sum of module header bytes 0-105, plus the sum of the bytes of the module

body 105. Bytes 110-127 (250) are for optional data.

Turning now to Figure 8, the steps for utilizing the invention are as follows: In step A a microcode update file in accordance with the teachings of the invention is constructed, preferably on computing device 80. The microcode update file may be in the form of data, for example, a font update, an update to a lookup table, etc., or may be in the form of a new executable program. The new executable program, referred to hereinafter as an executable, is in an object form (that is, the filename has the form " *.o") and can be any name. The executable preferably includes a procedure that allows the current microcode in the printer 10 to recognize the print job file 97 as a microcode update file including a new executable.

The microcode update file is configured as module body 105 and a corresponding module header 110 is constructed. The module header 110 and body 105 are placed in a print job file 97 as shown in step B. As mentioned above, the file header 100 of print job file 97 is constructed with information that identifies it as including a microcode update. The print job file is sent to the printer 10 in step C through any interface that is capable of receiving a print job file, preferably the I/O interface 75 (Figure 2), and downloaded into volatile memory 55.

In step D the current microcode controlling the processor 50 in the printer 10 recognizes the print job file 97 as a microcode update.

If the microcode update file includes a new executable there is also an option available to allow the new executable to download the remainder of the print job file 97, wherein the current microcode passes the pointers to the remainder of the print job file 97 to the new executable once it has been downloaded and is running. This option is designated in the file header 100 of the print job file 97 in byte 248 (195 in Figure 5). The option is defined by bits 0 and 1 of byte 248 and operates as follows:

As shown in step E, if bit 0 = 0, the option is not active, and all other bits are ignored. The print job file is recognized only as including a microcode update, and each module is treated as data. As such, each module is downloaded to volatile memory 55. As shown

in step F, the modules are copied into the storage device 63 and directory 64 (and subdirectory 65 if specified) designated in the respective module header 110 by bytes 60-100 (230 in Figure 7). A status page is printed (if not suppressed) including the upper and lower status page descriptors 145, 160 (see Figure 6). As shown in step G, if no errors occur, the modules in volatile memory 55 are erased and the modules in the storage device 63 and directory 64 are renamed if required. The processor 50 then reboots.

As shown in step H, if bit 0 = 1, the option is active and the first module header $110_1$ and body $105_1$ are downloaded into volatile memory 55, and the contents of the first module body $105_1$ are written into the storage area and directory 230 designated in the first module header $110_1$. In step I the current microcode then loads the new executable from the storage device 63 and directory 64 into execution memory 57 through an operating system call that enables dynamic downloading of new executables to execution memory 57. By dynamic downloading it is meant that the new executable may be loaded as the next task while the processor is performing another task. The operating system call also records and registers the names of any procedure that may be included in the new executable so it can be utilized by the current microcode if required. The new executable contains fixed name procedure calls that allow the current microcode to identify those procedures newly included in the new executable, once the new executable has been loaded into execution memory 57. Once loaded, the processor 50 transfers execution to the newly loaded executable and waits for it to return.

As shown in step J, if bit 1 = 0, when the operation of the new executable returns, the current microcode resumes control of the processor 50, and the current microcode will download the remainder of the print job file 97. A status page is printed (if not suppressed) including the upper and lower status page descriptors 145, 160. If the download is successful, the new executable is erased from non-volatile memory and the processor 50 reboots.

As shown in step K, if bit 1 = 1, the new executable will download the rest of the print job file 97 during its operation. In this case the new executable preferably includes additional code to process the remainder of the print job file 97. At this point, the new executable may perform any function that is capable of being programmed, including

writing new microcode to non-volatile memory and rebooting. However, if the new executable ends operations and the current microcode resumes control of the processor 50, a status page is printed (if not suppressed) including the upper and lower status page descriptors 145, 160, and if there are no errors, the new executable is erased from non-volatile memory and the processor 50 reboots.

Although described above in the context of specific computer peripherals, I/O interfaces, printer architectures, specific numbers of data fields, lengths of data fields, and specific arrangements of data fields in the print job file 97, and the various headers 100, 110 contained therein, those skilled in the art should appreciate that these are exemplary and indicative of presently preferred embodiments of these teachings, and are not intended to be read or construed in a limiting sense upon these teachings.